Final version by the authors. This paper was published as:

Verena Blaschke, Thora Daneyko, Jekaterina Kaparina, Zhuge Gao, and Johannes Dellert (2024).
"Navigable atom-rule interactions in PSL models enhanced by rule verbalizations, with an application to etymological inference."
In *Inductive Logic Programming (ILP 2022)*, pp. 15–24. Cham. Springer Nature Switzerland. DOI: 10.1007/978-3-031-55630-2_2

Navigable atom-rule interactions in PSL models enhanced by rule verbalizations, with an application to etymological inference

Verena Blaschke[0000−0002−1082−2740], Thora Daneyko[0000−0002−4079−0559], Jekaterina Kaparina[0000−0002−1898−0532], Zhuge Gao[0000−0003−1310−4805], and Johannes Dellert[0000−0002−9661−0668]

Seminar für Sprachwissenschaft, Eberhard Karls Universität Tübingen, Germany
{verena.blaschke,johannes.dellert}@uni-tuebingen.de

Abstract. Adding to the budding landscape of advanced analysis tools for Probabilistic Soft Logic (PSL), we present a graphical explorer for grounded PSL models. It exposes the structure of the model from the perspective of any single atom, listing the ground rules in which it occurs. The other atoms in these rules serve as links for navigation through the resulting rule-atom graph (RAG). As additional diagnostic criteria, each associated rule is further classified as exerting upward or downward pressure on the atom's value, and as active or inactive depending on its importance for the MAP estimate.
Our RAG viewer further includes a general infrastructure for making PSL results explainable by stating the reasoning patterns in terms of domain language. For this purpose, we provide a Java interface for "talking" predicates and rules which can generate verbalized explanations of the atom interactions effected by each rule. If the model's rules are structured similarly to the way the domain is conceptualized by users, they will receive an intuitive explanation of the result in natural language.
As an example application, we present the current state of the loanword detection component of EtInEn, our upcoming software for machine-assisted etymological theory development.

Keywords: PSL · Explainable reasoning · Visual model inspection

1 Introduction

Probabilistic Soft Logic (PSL) [1] is a popular templating language for a tractable class of graphical models which has already been applied to a wide range of domains, but poses some challenges in terms of understanding and debugging inference behavior. In this paper, we describe and publicly release additional tooling which we found necessary in order to address these challenges while developing a range of complex PSL models for tasks in historical linguistics.

We first describe our strategies for analyzing MAP states of PSL programs (Section 2), then introduce our approach to explaining inference behavior through rule verbalization (Section 3), outline the main features of our implementation (Section 4), and finally illustrate the application within our domain (Section 5).

2 Analyzing PSL Programs as Rule-Atom Graphs

A grounded PSL program consists of rules which define constraints and loss potentials on combinations of probabilistic atoms. Logical rules are disjunctions of literals whose value is defined via the strong disjunction of Łukasiewicz logic, whereas arithmetic rules directly express constraints on weighted sums of atom values. PSL facilitates the definition of large models by providing a first-order templating language where atoms are expressed as predicate symbols with arguments, and variables can be used to specify the combinations of atoms that can instantiate a rule template. The rules which make up the model are then produced by grounding the templates against a universe of ground atoms.

This flexibility comes at the cost of making the behavior of PSL models difficult to understand, to debug and to analyze. The grounding process makes the link between specification and model quite indirect, and MAP inference on the grounded model is much less reducible to individual reasoning steps than in classical programming paradigms, because messages will be propagated through many atoms in sometimes very complex ways to arrive at the MAP state.

While debugging for query-based probabilistic logic paradigms can rely on provenance graphs [8], a grounded PSL program is inspected most naturally via its **rule-atom graph (RAG)**. The RAG is a bipartite graph where each ground rule instance corresponds to a **rule node**, each ground atom in the universe is represented by an **atom node**, and the node for each rule is connected to the nodes for each of the atoms which occur in it.

VMI-PSL [6], a recent PSL debugging and analysis tool, can be seen as operating on the RAG, focusing on individual rules and their groundings as the primary points of interest. Whereas VMI-PSL mainly supports a debugging paradigm where rules are analyzed in terms of the number of groundings as well as aggregate measures of (dis)satisfaction, our tool focuses the analysis on the perspective of the individual ground atom whose value in a MAP state might be unexpected or undesired. Adopting this view of the structure of a PSL problem leads us to two diagnostic criteria which, to our knowledge, have only been discussed implicitly in the literature, and for which no tools have been available.

2.1 Upward and Downward Pressure

The inference goal in a feasible grounded instance of a PSL program can be expressed as minimizing the distance to satisfaction of its rules. Framed in terms of the rule-atom graph, this can be seen as applying pressure to the rule nodes, which in turn results in upward or downward pressures on the values of various atoms when being distributed throughout the graph. A MAP state for the PSL problem can thus be interpreted as a state of equilibrium between these pressures in the RAG, at which the weighted sum of hinge-loss potentials cannot be minimized any further by modifying the values assigned to atoms.

One of the interesting properties of PSL is that each rule-atom link can be classified as exerting upward or downward pressure on the value of the atom. For

example, if a grounding of the following example rule from the PSL documentation is unsatisfied, it exerts upward pressure on the value of `Knows(P1,P2)`, but downward pressure on the values of `Lives(P1,L)` and `Lives(P2,L)`:

```
20: Lives(P1,L) & Lives(P2,L) & (P1 != P2) -> Knows(P1,P2) ^2
```

This makes associated groundings of this rule relevant if we are interested in finding out why some grounding of `Knows(P1,P2)` receives a higher value than expected, but irrelevant if we are interested in the pressures pushing up the value of some grounding of `Lives(P1,L)`. To connect this perspective to the underlying mathematical structures, we can use the inequality representation of logical rules which forms the basis of the corresponding hinge-loss potential:

$$1 - \sum_{i \in I^+} y_i - \sum_{i \in I^-} (1 - y_i) \leq 0$$

A rule exerts upward pressure on exactly those atoms which correspond to variables y_i where $i \in I^+$, and downward pressure on those atoms corresponding to y_i where $i \in I^-$. In arithmetic rules, the direction of the pressure arises trivially from the sign of the atom's weight and the direction of the inequality.

2.2 Active and Inactive Rules

As in VMI-PSL, the distance to satisfaction of ground rules plays a central role in our debugging paradigm. We see ground rules which are unsatisfied in a MAP state as a key means to detect the places where the rule system is under pressure, i.e. where competing evidence might exist. In addition to making rule nodes sortable by distance to satisfaction, we also analyse their relevance.

If a ground rule has a distance to satisfaction of ≥ 0, this means that the rule is contributing to the value of the associated atoms, in the sense that if the corresponding potential were not part of the hinge-loss function, the function could be minimized further and the current estimate would cease to be a MAP state. In contrast, a negative distance to satisfaction implies that the value of the associated atoms in the MAP state would not be any different if the rule did not exist, making it irrelevant for understanding why the MAP state is optimal.

This difference can be framed in terms of rule activity. As a minor complication, the values assigned to atoms by MAP inference in the PSL reference implementation are not very precise. Rounding errors often lead to minor distances to satisfaction that cannot actually be interpreted as meaning that the rule is not satisfied. This leads us to a more robust counterfactual definition of activity: a ground rule is **active** at some state if its distance to satisfaction would reach some ϵ (by default, $\epsilon = 0.01$) if we changed the value of the context atom by 2ϵ in the potentially pressure-inducing direction, and **inactive** otherwise.

We have found rule activity to be a very useful diagnostic criterion for detecting problems in complex rule systems. If many groundings of some rule turn out to be inactive, this might indicate that its reasoning pattern is already covered by (a combination of) other rules, or that the rule needs to be tightened in some way in order to increase its influence on the results.

3 Verbalization of Atoms and Rules

In addition to its role as a debugging tool for complex PSL models, our second goal for the RAG viewer is the ability to communicate the system's reasoning back to the user. Unlike in some scenarios where the user will be content with a rough explanation in terms of the information that was used to infer a value [4], we want to generate explanations directed at a domain expert user who wants to be able to fully understand the system's reasoning, which is closer to the concept of explanation in symbolic expert systems [7].

Using terminology from the field of recommender systems [5], the rule-atom graph already provides an explicit model of the relationships between knowledge objects, and the decisive input values for a context atom are the ones which are connected to it though active rule nodes. An explanation could thus be provided by a direct visualization of the rule-atom graph where rule nodes are annotated by our diagnostic criteria, but as Kouki et al. [4] find for their PSL-based application, users prefer textual explanations over visualizations.

As an additional advantage, textual explanations make it easier to provide further inference and domain knowledge. Since PSL rule templates are powerful enough to translate common domain-specific reasoning patterns rather directly, the roles of many rules in the overall model can be grasped quite intuitively by a domain expert. For explainable inference, this leads to the idea of exposing the RAG, but verbalizing the mechanics of each ground rule in terms of domain-specific natural language.

In practice, we begin each explanation with an introductory sentence summarizing the reasoning pattern expressed by the rule. For our example rule, this could be: "If there is evidence that two people live at the same address, this makes it more likely that they know each other." A second sentence then fills in the specifics of the grounding. Since in our paradigm, the ground rules always occur from the perspective of some context atom, a ground rule is verbalized in different ways depending on which atom is currently being inspected.

Consider a grounding of our example rule where `P1 = "Bob"`, `P2 = "Alice"`, and `L = "18"` ("house number 18"). The shape of the second sentence depends on the role of the context atom within the rule. If the context atom is in the consequent of a logical rule (e.g. `Knows("Alice","Bob")`), the explanation will typically correspond directly to the intuition expressed by the rule: "We are certain that Bob lives in house number 18, and Alice probably lives in house number 18 as well, therefore Bob and Alice probably know each other." If the context atom occurs in the antecedent of a logical rule (e.g. `Lives("Alice","18")`), we explain that the values of the other atoms prevent it from taking a higher/lower value: "We are certain that Bob lives in house number 18, but there is some reason to doubt that Alice and Bob know each other, therefore we cannot be certain that Alice also lives in house number 18."

Special treatment is given to several boundary cases where a relevant atom takes an extreme value (0 or 1), e.g. by expressing that the rule is trivially satisfied, and providing verbalizations of the associated atoms in parentheses.

4 Implementation

4.1 Infrastructure

Our implementation[1] is built on a custom abstraction layer which we implemented in pure Java on top of the PSL reference implementation. This layer is publicly released as a Java package[2] which, among other features, provides an API for defining and administering PSL problems in Java (e.g. in order to generate problem instances programmatically), and to extract rule-atom graphs from the underlying PSL implementation. Our RAG viewer is implemented as a JavaFX component which allows to analyse the output of any PSL problem defined via our abstraction layer. In addition to providing a more reactive interface, this also means that unlike web-based services like VMI-PSL, our viewer runs locally without any server availability or data security issues.

4.2 Design of the RAG Viewer

The main functionality of the viewer can be described as an atom browser where for each atom, the reasoning leading to its value in the PSL output is laid out on a separate page. Each page contains representations of all the ground rules which had an influence on the atom's belief value, and these representations contain hyperlinks which allow to navigate to the other atoms tied together by that rule instance. The associated ground rules and constraints are divided into two blocks depending on the direction of pressure they exert on the context atom, and sorted by their weighted distance to satisfaction at the current MAP estimate, where inactive rules are greyed out. A sidebar provides the option to filter a list of all ground atoms by their argument constants, as well as sorting the matching atoms by their values in the MAP state.

4.3 Interface for Verbalizations

Our infrastructure also provides an interface and various helper methods for defining application-specific atom and rule verbalizations. A **talking atom** can express the associated predicate and its arguments as a noun phrase or full sentence. The value of the atom in the current solution can additionally be expressed as an adverb ("perhaps", "probably", "very likely") or, in some contexts, as a full phrase ("reaches a high level of certainty"). The interface for what we call a **talking rule** requires the model developer to write a method that, given the atom groundings and values, produces an explanation of the pressure exerted on the context atom. Good verbalizations will be highly dependent on the rule and the domain, but generic implementations are supported by template methods for the format and the different scenarios discussed in the previous section.[3]

[1] https://github.com/verenablaschke/psl-ragviewer
[2] https://github.com/jdellert/psl-infrastructure
[3] See https://github.com/jdellert/psl-infrastructure/wiki/Rule-verbalization for more details.

5 Application Example

We illustrate the concepts introduced in the previous sections in the context of our application of PSL within the framework of EtInEn, our forthcoming software suite for machine-assisted historical linguistics. Historical linguists try to explain the origin and development of attested languages by forming theories about their common ancestors. Language families arise from repeated language splits followed by divergence of the descendant languages. In addition to this tree-shaped process, languages will often copy material from languages that they are in contact with, a process that is called **borrowing**. On the level of the lexicon, this will result in **loanwords** entering the language. Layers of loanwords often provide important hints about the nature of (pre)historic cultural contacts.

5.1 Etymological Reasoning

Establishing an **etymology** means to trace the origin of a word into the past, thereby explaining which other words (most often in other languages) are related to the word by common descent. This entails the important step of deciding for **homologues** (words that are etymologically related) whether they are a result of inheritance or borrowing. Many different sources of evidence need to be brought to bear in order to find loanwords. Our example model derives judgments from two types of patterns. The first is to project homologue set membership upwards in the family tree, detecting situations where the homologue set being used for a concept in some language is different from the one reconstructed for the parent language, e.g. in a situation where a single language uses a word from a different family branch or family. The second type of evidence, which helps to detect loanwords within a homologue set, is based on deviations from the expectation that the phonetic similarity of words will roughly mirror language relatedness.

5.2 Expressing Etymologies in PSL

The following predicates are used to express the input data and the etymologies:

`Einh(X,Y)`	belief that the etymology of X is inheritance from Y
`Eloa(X,Y)`	belief that the etymology of X is borrowing from Y
`Eunk(X)`	belief that the etymology of X is unknown (outside scope of data)
`Fhom(X,H)`	belief that the word X belongs to the homologue set H
`Fsim(X,Y)`	phonetic similarity of the words X and Y
`Xinh(X,Y)`	possibility of inheritance (language of Y is parent of that of X)
`Xloa(X,Y)`	possibility of borrowing (language of Y influenced that of X)

In order to create the universe for a specific problem, the language tree and the directional language contacts are translated into `Xinh` and `Xloa` observations as well as the corresponding `Einh` and `Eloa` targets. Attested words have observed `Fhom` and `Fsim` values, potentially derived by specialized algorithms, the remaining atoms are targets and will be inferred in conjunction with the values of the `Einh(X,Y)` and `Eloa(X,Y)` items that constitute the model output.

5.3 The Etymological Inference Model

The constraints mainly ensure symmetry and a certain degree of transitivity, and that homologue judgments and etymologies for each word form distributions:

```
Eloa(X,Y) + Eloa(Y,X) <= 1 .
Einh(X,+Y) + Eloa(X,+Z) + Eunk(X) = 1 .
Einh(X,Y) & Fhom(Y,H) -> Fhom(X,H).
Eloa(X,Y) & Fhom(Y,H) -> Fhom(X,H).
Fhom(X,+H) = 1.
Fsim(X,Y) = Fsim(Y,X) .
Fsim(X,Y) & Fsim(Y,Z) & (X != Y) & (X != Z) & (Y != Z) -> Fsim(X,Z) .
```

`Eloa` and `Eunk` atoms carry negative priors, the remaining rules are:

a. `2.0: Einh(X,Z) & Einh(Y,Z) & (X != Y) -> Fsim(X,Y)`
b. `1.0: Fsim(X,Y) & Einh(X,W) & Einh(Y,Z) & (W != Z) -> Fsim(W,Z)`
c. `1.0: Eloa(X,Y) & Fsim(X,Z) & (Y != Z) & (X != Z) -> Fsim(X,Y)`
d. `0.6: Fhom(X,H) & Xinh(X,Z) -> Fhom(Z,H)`
e. `0.2: Fhom(Z,H) & Xinh(X,Z) -> Fhom(X,H)`
f. `0.4: Fhom(X,H) & Fhom(Y,H) & Xinh(X,Y) -> Einh(X,Y)`
g. `1.0: Fhom(X,H) & ~Fhom(Y,H) & Xinh(X,Y) & Xloa(X,Z) -> Eloa(X,Z)`

To state the intuitions behind each of these rules in the given order, there is a very strong preference for two words which are inherited from the same word to be phonetically similar *(a),* and sources of similar words should be similar as well *(b).* A borrowed word should be more similar to its source than to any other word *(c).* Evidence of homologue set presence is propagated along parent-child links, with the child-to-parent direction being dominant *(d, e).* If parent and child share a homologue set, that suggests inheritance *(f).* In contrast, if there is any reason to doubt the presence of a homologue set in the parent, an available loanword etymology becomes much more likely *(g).*

5.4 Exploring an Example Instance

In our example, we want to infer that English *take* is a Norse loan (i.e. from Proto-North-Germanic) based on its equivalents in seven modern Germanic languages. Our Germanic tree consists of the West Germanic branch with three children (German, Dutch, and Old English with the attested child English) and the North Germanic branch with children West Scandinavian (Icelandic and Norwegian), Swedish, and Danish. This tree is extended into a network by contact links from German to every modern North Germanic language except Icelandic, from Danish to Norwegian, and from Proto-North-Germanic to Old English.

The input data consist of the following equivalents of TAKE: German *nehmen*, Dutch *nemen*, English *take*, Danish *tage*, Norwegian *ta*, Swedish *ta*, Icelandic *taka*. The `Fsim` and `Fhom` values for these forms are derived by applying the IWSA algorithm [2] to the phonetic data from the NorthEuraLex database [3].

Inference is successful. Below, we show two screenshots of our RAG browser displaying an explanation for an etymological reasoning step. A linguist user who wants do understand the output will navigate through the relevant atoms by following the links inside the verbalizations.

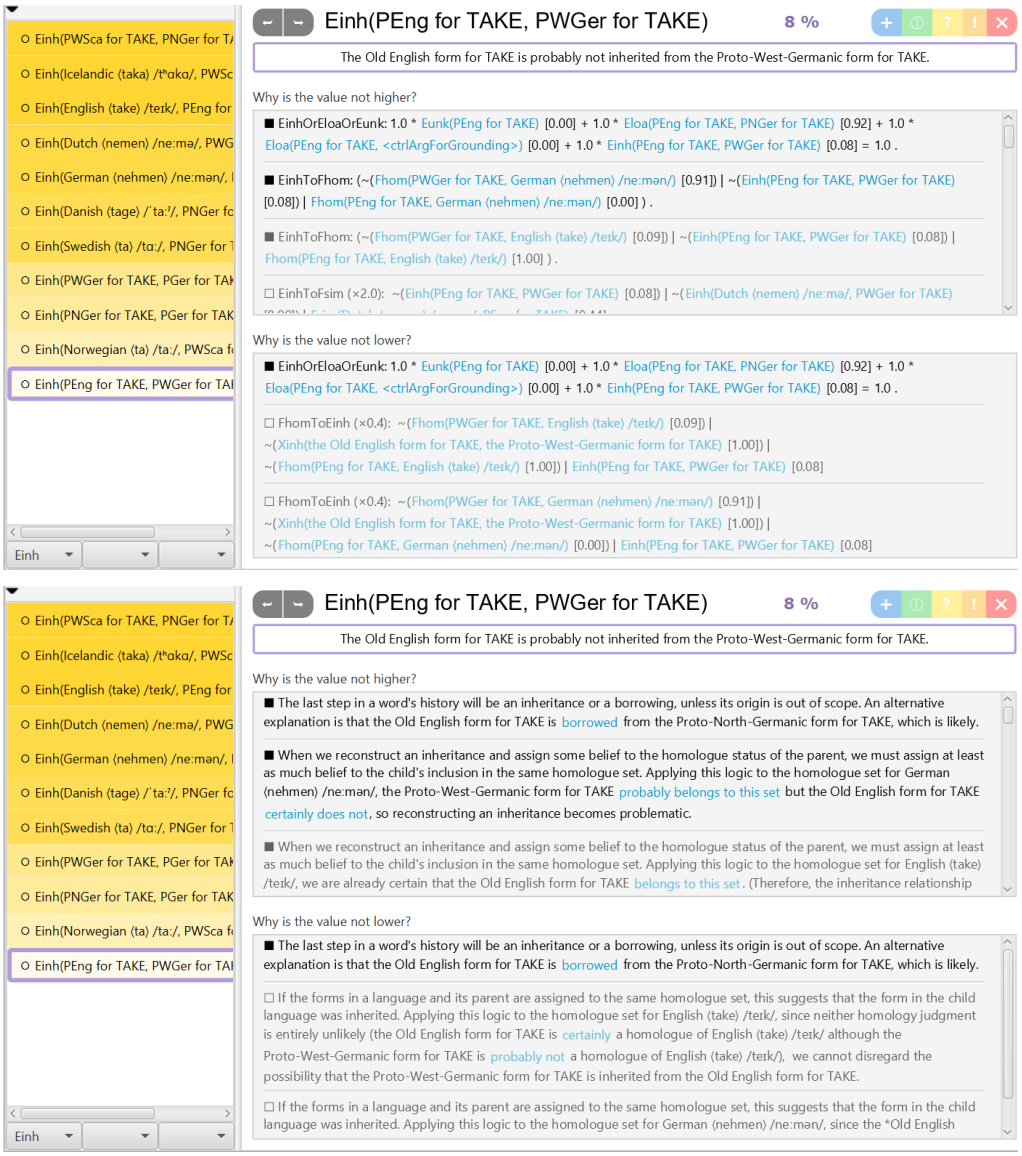

Fig. 1. Our tool explains why the Old English word for TAKE is probably not inherited from Proto-West-Germanic. Users can toggle between the internal representation of the ground rules and the verbalized explanations. Inactive rules have been greyed out.

6 Conclusion

In addition to the functionality described in this paper, several experimental features that will be of interest to other PSL users are still in development. They include the ability to perform conditional inferences after fixing the values of selected atoms, and support for persisting RAGs in files for post-hoc inspection.

We also plan to extend the user interface by the possibility to switch to a rule-focused view in order to receive an overview of all groundings, but enhanced by our diagnostic criteria, such as activity status at the MAP state.

On the application side, our loanword detection module is being expanded by additional sources of evidence, such as the general tendency for words for concepts from certain cultural or technological spheres to be borrowed jointly. In the finished EtInEn architecture where soundlaws and reconstructions will be modeled, the very important criterion of deviation from sound laws is going to contribute to loanword inference as well.

Acknowledgements This work has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (CrossLingference, grant agreement no. 834050) as well as the Institutional Strategy of the University of Tübingen (Deutsche Forschungsgemeinschaft, ZUK 63) and a RiSC grant by the MWK Baden-Württemberg.

References

1. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss Markov random fields and Probabilistic Soft Logic. Journal of ML Research **18**(109), 1–67 (2017)
2. Dellert, J.: Combining information-weighted sequence alignment and sound correspondence models for improved cognate detection. In: Proceedings of COLING 2018. pp. 3123–3133. Association for Computational Linguistics (2018)
3. Dellert, J., Daneyko, T., Münch, A., Ladygina, A., Buch, A., Clarius, N., et al.: NorthEuraLex: a wide-coverage lexical database of northern Eurasia. Language Resources and Evaluation **1**(54), 273–301 (2020). https://doi.org/10.1007/s10579-019-09480-6
4. Kouki, P., Schaffer, J., Pujara, J., O'Donovan, J., Getoor, L.: Personalized explanations for hybrid recommender systems. In: Proceedings of IUI 2019. pp. 379–390 (2019). https://doi.org/10.1145/3301275.3302306
5. Nunes, I., Jannach, D.: A systematic review and taxonomy of explanations in decision support and recommender systems. User Modeling and User-Adapted Interaction (27), 393–444 (2017). https://doi.org/10.1007/s11257-017-9195-0
6. Rodden, A., Salh, T., Augustine, E., Getoor, L.: VMI-PSL: Visual model inspector for probabilistic soft logic. In: Proceedings of RecSys 2020. pp. 604–606 (2020). https://doi.org/10.1145/3383313.3411530
7. Southwick, R.W.: Explaining reasoning: an overview of explanation in knowledge-based systems. The Knowledge Engineering Review **6**(1), 1–19 (1991). https://doi.org/10.1017/S0269888900005555
8. Wang, S., Lyu, H., Zhang, J., Wu, C., Chen, X., Zhou, W., et al.: Provenance for probabilistic logic programs. In: Proceedings of EDBT 2020 (2020). https://doi.org/10.5441/002/edbt.2020.14